Précision numérique dans le cumul d'un grand nombre de termes

Michèle Raphalen, Bernard Philippe

IRIA

CENTRE DE RENNES
IRISA

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tel (3) 954 90 20

Rapports de Recherche

N° 412

PRÉCISION NUMÉRIQUE DANS LE CUMUL D'UN GRAND NOMBRE DE TERMES

Michèle RAPHALEN
Bernard PHILIPPE

Mai 1985

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTÈMES ALÉATOIRES

Campus Universitaire de Beaulieu
Avenue du Général Leclerc
35042 - RENNES CÉDEX
FRANCE
Tél. : (99) 36.20.00
Télex : UNIRISA 95 0473 F

Publication Interne n° 253

PRECISION NUMERIQUE DANS LE CUMUL D'UN GRAND NOMBRE DE TERMES

Michèle RAPHALEN
Bernard PHILIPPE

Avril 1985
44 pages

Résumé : Nous comparons la précision obtenue dans le cumul d'un grand nombre de réels, pour deux algorithmes. Les erreurs commises dans le cumul sont modélisées par des variables aléatoires.

Abstract : We compare the accuracy of two particular algorithms for the addition of a large amount of reals. Round off errors are delt using random variables.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
(L.A. 227)
UNIVERSITÉ DE RENNES 1 I.N.S.A. DE RENNES

INSTITUT NATIONAL DE RECHERCHE
EN INFORMATIQUE ET EN AUTOMATIQUE
(LABORATOIRE DE RENNES)

INTRODUCTION

Pour résoudre des problèmes numériques de plus en plus importants, on est amené à augmenter la capacité des calculateurs ; mais parallèlement à cet accroissement des performances, il faut garantir une bonne précision dans les calculs.

Une solution consiste à utiliser le mot de 64 bits comme format standard pour un nombre flottant. Cette solution donne une précision satisfaisante dans la plupart des cas, mais est coûteuse en mise en oeuvre, tant au niveau des opérateurs flottants qu'au niveau de la mémoire.

Pour la plus grande partie des calculs, la précision obtenue en utilisant des mots de 32 bits serait suffisante. Il serait alors intéressant, pour un calculateur de moyenne puissance, de se limiter au mot de 32 bits pour un nombre flottant, avec la possibilité du calcul en double précision, coûteux en temps, mais peu fréquent. Cette solution conduit à améliorer la stabilité des algorithmes, c'est à dire à les rendre le plus insensible possible aux imprécisions de calcul.

Le but de cet article est de comparer deux méthodes de calcul pour le cumul d'un grand nombre de termes. Ce problème mérite une attention particulière, car beaucoup d'algorithmes le rencontrent, en particulier lorsqu'ils utilisent des produits scalaires sur de longs vecteurs.

Pour mesurer la précision sur le résultat, une première approche consiste à établir une majoration de l'erreur. Cette majoration n'est jamais mise en défaut, mais elle est en général pessimiste vis-à-vis de la qualité du calcul. Une deuxième approche consiste à donner une estimation de l'erreur, à partir d'un modèle probabiliste. La difficulté réside alors dans la construction d'un modèle fiable permettant de définir cette estimation.

On étudie ici l'erreur commise dans le calcul de la somme de n termes suivant une méthode en ligne et une méthode en arbre. Les erreurs sont modélisées par des variables aléatoires dont on calcule l'espérance et la variance. Pour chacune des deux méthodes, on distingue le cas où les n termes sont de même signe du cas où les n termes sont de signe variable. Dans le premier cas, on compare les erreurs relatives commises par les deux méthodes. Dans le second cas, le résultat du cumul pouvant être nul, on compare les erreurs absolues.

I. Arithmétique flottante

I.1. Nombre flottant normalisé

Un nombre flottant normalisé A à t bits de mantisse est défini en système binaire par son signe, son exposant, e, sa mantisse, m, de la manière suivante :

$$A = (m\ 2^{-t})2^{e} \qquad e \in \mathbb{Z},\ 2^{t-1} \leqslant m < 2^{t}$$

Pour tout réel x non nul, sa représentation flottante $\tilde{x}$ est définie par :

$$\tilde{x} = \text{signe}(x)\ (m\ 2^{-t})2^{e} \quad \text{avec } m - \frac{1}{2} \leqslant |x|\ .\ 2^{t-e} < m + \frac{1}{2}$$

Le nombre réel x est donc représenté par un nombre flottant suivant la règle de l'arrondi.

L'erreur absolue commise dans cette représentation est :

$$\varepsilon_a = x - \tilde{x} \qquad \varepsilon_a \in [-2^{e-t-1}\ ,\ 2^{e-t-1}[$$

L'erreur relative est alors majorée par $2^{-t}/(1-2^{-t})$.

Remarque :

- les réels de la forme : $y = (m + \delta)2^{e-t}$ avec $\delta \in [-1/2\ ,\ \frac{1}{2}[$ ont la même représentation flottante.

Pour modéliser la probabilité d'apparition d'une mantisse m à t chiffres binaires, on utilise les résultats de Tsao [TS 74] : il montre que pour t suffisamment grand, on peut considérer la partie fractionnaire $F = m\ 2^{-t}$ comme une variable aléatoire continue de densité de probabilité :

$$g_F(f) = \frac{1}{f \ln 2} \qquad \frac{1}{2} \leqslant f \leqslant 1$$

La fonction de répartition est alors :

$$\Pi\ [F < B] = 1 + \log_2 B \qquad \frac{1}{2} \leqslant B \leqslant 1$$

I.2. Opérations flottantes sur deux opérandes

I.2.1. addition, multiplication, division

Soit T l'opération effectuée sur les nombres flottants normalisés $\tilde{x}$ et $\tilde{y}$ à t bits de mantisse ; ces nombres sont de même signe dans le cas de l'addition.

La représentation flottante de xTy est définie par :

$$fl\ (\tilde{x} T \tilde{y}) = (\tilde{x} T \tilde{y})\ (1+\varepsilon) \text{ avec } \quad |\varepsilon| \leqslant 2^{-t}$$

La variable aléatoire ε suit la loi de probabilité (figure 1) décrite dans (TS 74) :

$$f_{\varepsilon}\ (\varepsilon_0) = \begin{cases} \dfrac{2^{t-1}}{\ln 2} & 0 \leqslant |\varepsilon_0| \leqslant 2^{-t-1} \\ \dfrac{1}{\ln 2}\ \left(\dfrac{1}{2|\varepsilon_0|} - 2^{t-1} \right) & 2^{-t-1} < |\varepsilon_0| \leqslant 2^{-t} \end{cases}$$

C'est une variable centrée, de variance :

$$\sigma^2 = \mathbb{E}\ (\varepsilon^2) = \frac{2^{-2t}}{8 \ln 2}$$

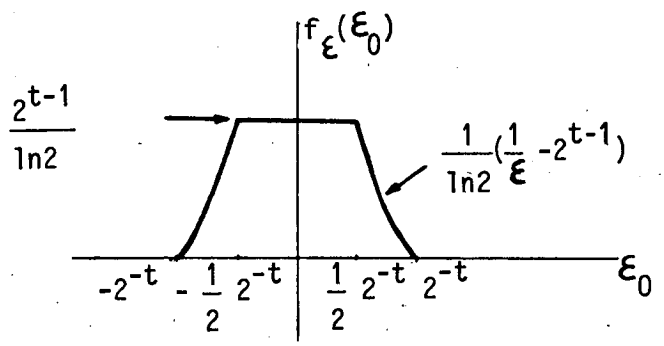


Figure 1 : Distribution de la variable ε
(cas du calcul avec arrondi).
(D'après (TS 74))

I.2.2. soustraction

Soit $\tilde{x} = (m_1\ 2^{-t}).2^{e_1}$

$\tilde{y} = (m_2\ 2^{-t}).2^{e_2}$ $\qquad 2^{t-1} \leqslant m_1, m_2 < 2^t, \quad e_1 > e_2$

La soustraction flottante $\tilde{x} - \tilde{y}$ se déroule de la manière suivante (Figure 2).

* dénormalisation de $\tilde{y}$: décalage à droite de $k = e_1 - e_2$ positions binaires de m_2. On obtient une mantisse m'_2

* soustraction de m'_2 à m_1. On obtient la mantisse m'_3

* normalisation du résultat (décalage à gauche de m positions).

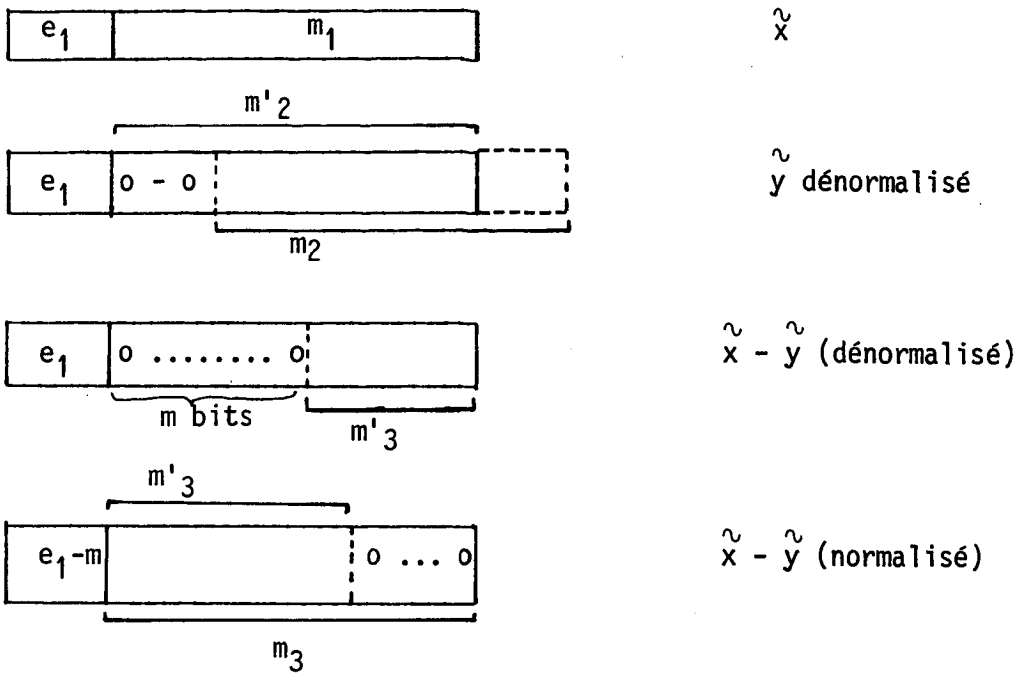


Figure 2 : Soustraction de 2 nombres

La quantité m donne une indication sur l'erreur commise, bien que dans certains cas celle-ci soit nulle (WI 63, p 9).

Dans (FE 82) , on trouve une mesure de la probabilité qu'il y ait une normalisation de m bits sachant que la différence d'exposant est k ; dans les hypothèses probabilistes définies au paragraphe (I.1) cette probabilité est donnée par la formule :

$$\mathbb{P}(k,m) = \frac{1}{(\ln 2)^2} \iint_{B(k,m)} \frac{dx}{x} \frac{dy}{y}$$

où $B(k,m) = \{(x,y) / \frac{1}{2} \leqslant x < 1 , \frac{1}{2} \leqslant y < 1, 2^{-(m+1)} \leqslant x - 2^{-k}y < 2^{-m}\}$

$\mathbb{P}(k,m)$ est nul pour $k \geqslant 2$ et $m \geqslant 2$.

Les valeurs de $\mathbb{P}(k,m)$ pour $k = 0$, $k = 1$ et $m = 0$, $m = 1$ sont données en figure 3.

m	$\mathbb{P}(o,m)$	$\mathbb{P}(1,m)$
0	0.00	22.14
1	24.75	50.00
2	30.80	21.25
3	20.38	4.98
4	11.55	1.22
5	6.13	.31
6	3.16	.08
7	1.60	.02
8	.82	
9	.41	
10	.20	
11	.10	
12	.05	
13	.03	
14	.01	
15	.01	
	100.00	100.00

k	$\mathbb{P}(k,o)$	$\mathbb{P}(k,1)$
2	55.77	44.23
3	76.15	23.85
4	87.56	12.44
5	93.64	6.36
6	96.78	3.21
7	98.38	1.62
8	99.19	.81
9	99.59	.41
10	99.80	.20
11	99.90	.10
12	99.95	.05
13	99.97	.03
14	99.99	.01
15	99.99	.01

Figure 3 : Tables de la distribution conditionnelle
P(k,m) : (en %) (d'après (FE 82))

I.3. Accroissement de la précision

Dans l'addition et la soustraction de deux nombres normalisés, des bits sont perdus lors de la dénormalisation. On aimerait garder cette information perdue pour la normalisation du résultat.

Pour obtenir un résultat correct, on peut définir un accumulateur double précision (WI 63). Cependant, deux bits de garde suffisent pour obtenir la même précision (KU 77).

En effet, pour l'addition, la normalisation éventuelle provoque un décalage vers la droite de une position donc le bit sortant détermine l'arrondi ; s'il n'y a pas de normalisation, un bit de garde est nécessaire pour déterminer cet arrondi.

Pour la soustraction, la normalisation peut introduire un nombre m quelconque de bits. Mais si $m \geqslant 2$, alors la dénormalisation du deuxième opérande peut provoquer un décalage vers la droite d'au maximum 1 position ($P(k,m) = 0$ pour $m \geqslant 2$ et $k \geqslant 2$). On a alors besoin de deux bits de garde, l'un pour le bit à introduire, l'autre pour l'arrondi.

II. Précision dans le calcul d'un cumul

On étudie ici l'erreur commise dans le cumul de n nombres réels (x_0, x_1,...,x_{n-1}) en arithmétique flottante.

II.1. Formalisation du problème

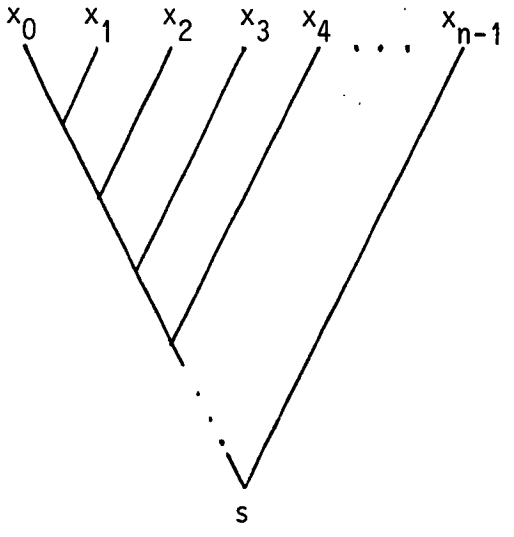
Le calcul de la somme :

$$s = \sum_{i=0}^{n-1} x_i$$

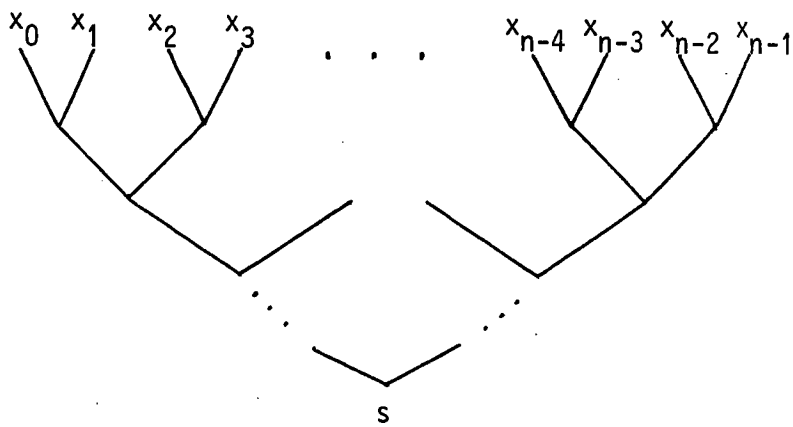
à l'aide d'opérateur flottant peut se faire suivant deux méthodes :

- méthode en ligne : $s=(...((x_0+x_1)+x_1)... + x_{n-2}) + x_{n-1}$
- méthode en arbre : $s=(...(((x_0+x_1)+(x_2+x_3)) + ((x_4+x_5)+(x_6+x_7)))+ ...)$

Nous appelons "méthode en ligne" la méthode conduisant à un arbre de calcul déséquilibré, par opposition à la "méthode en arbre" qui définit un arbre équilibré (symétrique). Ces deux méthodes sont décrites par les graphes suivants :



Méthode en Ligne



Méthode en Arbre

On étudie la précision obtenue par chacune de ces méthodes. Pour cela, on établit la relation entre l'erreur relative Λ sur le résultat (forward analysis (WI 63)) et les erreurs relatives η_i reportées sur les termes x_i de la somme (backward analysis).

Cette relation est donnée par :

$$fl\left(\sum_{i=0}^{n-1} x_i\right) = \left(\sum_{i=0}^{n-1} x_i\right)(1+\Lambda) = \sum_{i=0}^{n-1} x_i\,(1+\eta_i)$$

où $fl\left(\sum_{i=0}^{n-1} x_i\right)$ représente le résultat obtenu par l'une ou l'autre des méthodes.

Si $s \neq 0$ alors cette relation s'écrit :

$$1+\Lambda = \sum_{i=0}^{n-1} \alpha_i\,(1+\eta_i) \qquad \alpha_i = \frac{x_i}{s} \; ; \quad \sum_{i=0}^{n-1} \alpha_i = 1$$

Dans ce qui suit, les grandeurs Λ et η_i sont supposées être des variables aléatoires.

II.2. Cumul de termes de même signe

II.2.1. étude générale

Nous avons vu que toute addition de deux nombres flottants normalisés $\tilde{x}$ et $\tilde{y}$ introduit l'erreur relative ε telle que :

$$fl(x+y) = (x+y)\,(1+\varepsilon).$$

On évalue, pour chaque méthode, l'écart type de la v.a. $(1+\Lambda)$, qui est une mesure de la moyenne des écarts.

Pour cela, on calcule la quantité :

$$\mathbb{E}\left((1+\Lambda)^2\right) = \sum_{i=0}^{n-1} \alpha_i^2\, \mathbb{E}\left((1+\eta_i)^2\right) + 2 \sum_{0 \leqslant i < j \leqslant n-1} \alpha_i\,\alpha_j\, \mathbb{E}\left((1+\eta_i)(1+\eta_j)\right).$$

Le calcul de $\mathbb{E}\left((1+\Lambda)^2\right)$ se fait sous l'hypothèse suivante :

(H1) : Toutes les erreurs relatives apparaissant dans le calcul du cumul sont des v.a. indépendantes, de loi f_ε décrite au paragraphe I.2.1.

Ces erreurs sont occasionnées

- soit par la représentation flottante des réels x_i : ce sont les erreurs (ε').
- soit par l'addition flottante de deux opérandes : ce sont les erreurs (ε).

La variance $V(1+\Delta)$ est un polynôme $\mathcal{P}$ de variable a :

$$a = 1 + V(\varepsilon) = 1 + \sigma^2$$

Le polynôme $\mathcal{P}$ s'annule pour a = 1. On peut donc écrire :

$$\mathcal{P}(a) = (a-1)\,\mathcal{P}'(1) + 0\,(a-1)^2$$

Dans ce qui suit, $\tilde{x}_i$ désigne la représentation flottante du réel x_i : $\tilde{x}_i = x_i(1 + \varepsilon'_i)$.

1- méthode en ligne

$s_0 = \tilde{x}_0$

$s_r = fl(s_{r-1} + \tilde{x}_r) = (s_{r-1} + \tilde{x}_r)(1 + \varepsilon_r)$ $\quad r = 1\ ,\ n-1$

d'où :

$$1 + \eta_0 = (1 + \varepsilon'_0) \prod_{j=1}^{n-1} (1 + \varepsilon_j)$$

$$1 + \eta_i = (1 + \varepsilon'_i) \prod_{j=i}^{n-1} (1 + \varepsilon_j) \qquad i=1,\ n-1$$

$$\mathbf{E}\left((1 + \eta_0)^2\right) = a^n$$

$$\mathbf{E}\left((1 + \eta_i)^2\right) = a^{n-i+1} \qquad i=1,\ n-1$$

$$\mathbf{E}\left((1 + \eta_i)(1 + \eta_j)\right) = a^{n-j} \qquad 0 \leqslant i < j \leqslant n-1$$

On en déduit :

$$\mathcal{P}(a) = \alpha_0^2\, a^n + \sum_{i=1}^{n-1} \alpha_i^2\, a^{n-i+1} + 2 \sum_{0 \leqslant i < j \leqslant n-1} \alpha_i\, \alpha_j\, a^{n-j} - 1$$

$$\boxed{\mathcal{P}'(1) = n\,\alpha_0^2 + \sum_{i=1}^{n-1} (n-i+1)\,\alpha_i^2 + 2 \sum_{0 \leqslant i < j \leqslant n-1} (n-j)\,\alpha_i\,\alpha_j}$$

2- méthode en arbre

On suppose ici : $n = 2^p$

$s_0^i = \tilde{x}_i \qquad i = 0,\ n-1$

$s_r^i = fl(s_{r-1}^{2i} + s_{r-1}^{2i+1}) = (s_{r-1}^{2i} + s_{r-1}^{2i+1})(1 + \varepsilon_r^i)$ $\quad r = 1,\ p$

$\quad i = 0,\ \dfrac{n}{2^r} - 1$

d'où : $1 + \eta_i = (1 + \varepsilon'_i) \prod_{j=1}^{p} (1 + \varepsilon_j^{ij})$ $\quad i=0, n-1$

où les ε_j^{ij} sont les erreurs engendrées par les additions (noeuds) rencontrées sur le chemin menant de $\tilde{x}_i$ à s_p^o (figure 4).

Exemple : p = 3

Le chemin menant de $\tilde{x}_2$ à s_3^o passe par : s_1^1, s_2^o, s_3^o. On a donc :

$1 + \eta_2 = (1 + \varepsilon'_2)\ (1 + \varepsilon_1^1)\ (1 + \varepsilon_2^o)\ (1 + \varepsilon_3^o)$

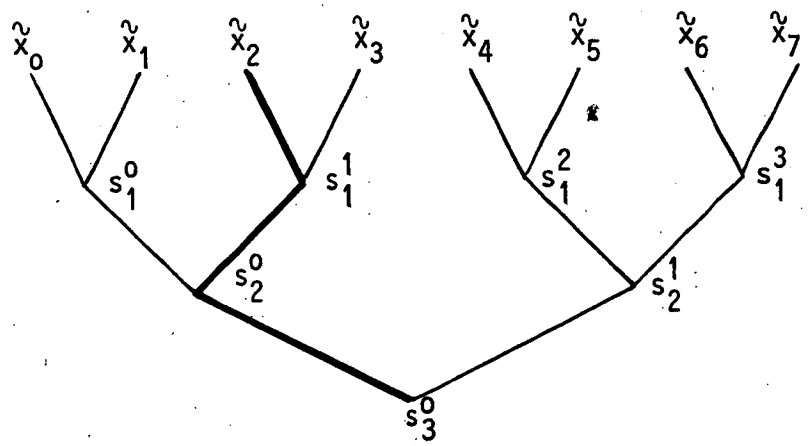


Figure 4 : Schéma de calcul de l'addition en arbre (p = 3).

$$\mathbb{E}\left[(1 + \eta_i)^2 \right] = a^{p+1} \qquad i = 0\ ,\ n-1$$

$$\mathbb{E}\left[(1 + \eta_i)\ (1 + \eta_j) \right] = a^{\delta} \qquad 0 \leqslant i < j \leqslant n-1$$

où $\delta = \phi(i,j)$ est le nombre de noeuds communs dans les chemins menant de $\tilde{x}_i$ à s_p^o et de $\tilde{x}_j$ à s_p^o ($i < j$).

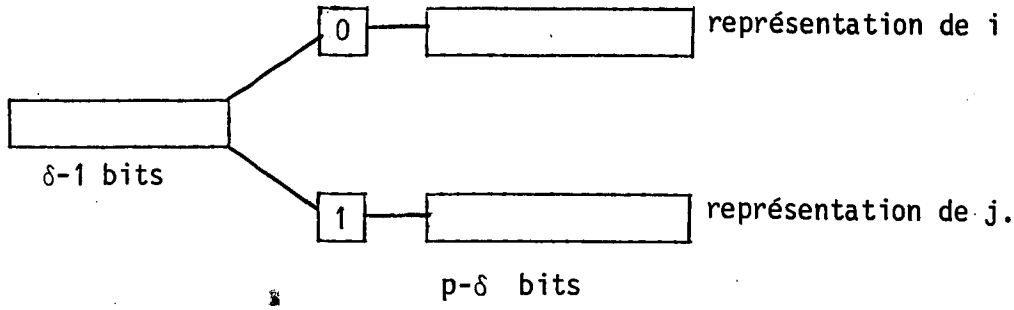
On en déduit :

$$\mathcal{P}(a) = \left(\sum_{j=0}^{n-1} \alpha_j^2 \right) a^{p+1} + 2 \sum_{\delta=1}^{p} \left(\sum_{(i,j)\in \phi^{-1}(\delta)} \alpha_i\, \alpha_j \right) a^{\delta} - 1$$

$$\boxed{\mathcal{P}'(1) = (p+1) \sum_{j=0}^{n-1} \alpha_j^2 + 2 \sum_{\delta=1}^{p} \delta \left(\sum_{(i,j)\in\phi^{-1}(\delta)} \alpha_i\, \alpha_j \right)}$$

Remarques :

(i) Si on part de la racine s_p^o , les chemins correspondants à $\tilde{x}_i$ et $\tilde{x}_j$ sont confondus sur δ étapes ssi les $(\delta - 1)$ bits de poids fort de i et j sont identiques, le $\delta^{\text{ème}}$ bit étant différent (c.a.d. 0 pour i, 1 pour j, car $i < j$).



on a donc :

$$\text{card}(\phi^{-1}(\delta)) = 2^{\delta-1} * (2^{p-\delta})^2 = 2^{2p-\delta-1}.$$

(ii) Le changement d'indice : $i \to n-i-1$ donne le même résultat de cumul. En effet, les arbres obtenus sont symétriques et l'addition flottante est commutative. Cette invariance n'existe pas dans la méthode en ligne.

On étudie maintenant deux applications :
- les termes du cumul sont de même grandeur
- les termes du cumul sont en progression géométrique.

Dans ce qui suit, $V^l(1 + \Lambda)$ (resp. $V^a(1 + \Lambda)$) désigne la variance de l'erreur commise par la méthode en ligne (resp. en arbre).

II.2.2. application au cas de termes de même grandeur

Dans ce cas, on approche le coefficient α_i par la constante $\frac{1}{n}$.

1. méthode en ligne

$$\mathcal{P}'(1) = \frac{n}{n^2} + \sum_{i=1}^{n-1} \frac{n-i+1}{n^2} + 2 \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} \frac{n-j}{n^2}$$

$$= \frac{2n^3 + 3n^2 + 7n - 6}{6n^2}$$

d'où :

$$\boxed{V^l(1 + \Lambda) = \left(\frac{n}{3} + \frac{1}{2} + \frac{7}{6n} - \frac{1}{n^2}\right) \sigma^2 + O(\sigma^4)}$$

2.méthode en arbre

$$\mathcal{P}'(1) = \frac{n\,(p+1)}{n^2} + \frac{2}{n^2} \sum_{\delta=1}^{P} \delta\, 2^{2p-\delta-1}$$

$$= \frac{(p+1)}{n} + \sum_{\delta=1}^{P} \delta\, 2^{-\delta}$$

$$= \left(2 - \frac{1}{n}\right)$$

d'où :

$$\boxed{V^a\,(1+\Lambda) = \left(2 - \frac{1}{n}\right)\,\sigma^2 + 0\,(\sigma^4)}$$

3.comparaison des deux méthodes

Le rapport r_σ de l'écart-type de l'erreur relative commise par la méthode en ligne par l'écart-type de l'erreur relative commise par la méthode en arbre vaut :

$$r_\sigma \sim \sqrt{n/6}$$

La courbe donnant r_σ en fonction de n est donnée à l'annexe 1. Dans le cas de termes de même grandeur, la précision obtenue par la méthode en arbre est meilleure que celle obtenue par la méthode en ligne.

II.2.3. application au cas de termes en progression géométrique

Dans ce cas, on a :

$x_i = kq^i$, $\alpha_i = q^i\,\frac{q-1}{q^n-1}$ où $q > 0$ et k quelconque.

1.méthode en ligne

$$\mathcal{P}'(1) = \left(\frac{q-1}{q^n-1}\right)^2 \left[\sum_{i=0}^{n-1} (n-i)\, q^{2i} + \sum_{i=1}^{n-1} q^{2i} + 2 \sum_{i=0}^{n-2} q^i \sum_{j=i+1}^{n-1} (n-j) q^j \right]$$

$$= \frac{2q^{2n+2} - 2q^{2n+1} + q^{2n} - 2q^{n+2} - 2q^{n+1} - q^4 + 2q^3 + nq^2 + 2q - n}{(q^n-1)^2\,(q^2-1)}$$

On distingue à ce niveau trois cas, suivant les valeurs de q.

- cas 1 : $q \gg 1$.

$$\mathcal{S}'(1) = 2 - \frac{2}{q} + 0\left(\frac{1}{q_2}\right)$$

On en déduit :

$$\boxed{V^1(1+\Lambda) = \sigma^2\left(2-\frac{2}{q}\right) + 0\left(\frac{\sigma^2}{q^2}\right) + 0\left(\sigma^4\right)}$$

- cas 2 : $0 < q \ll 1$

$$\mathcal{S}'(1) = n - 2q + q^4 + 0(q^5)$$

On en déduit :

$$\boxed{V^1(1+\Lambda) = n\sigma^2 - \sigma^2\left(2q - q^4\right) + 0(\sigma^2 q^5) + 0(\sigma^4)}$$

- cas 3 : $q = 1 + h \quad |h| \ll 1$

Le numérateur et le dénominateur de $\mathcal{S}'(1)$ s'annulent en $q = 1$, ainsi que leurs dérivées premières et secondes. Par le rapport des dérivées troisièmes, on obtient :

$$\mathcal{S}'(1) = \frac{n}{3} + \frac{1}{2} + \frac{7}{6n} - \frac{1}{n^2} + 0(h)$$

On en déduit :

$$\boxed{V^1(1+\Lambda) = \sigma^2\left(\frac{n}{3} + \frac{1}{2} + \frac{7}{6n} - \frac{1}{n^2}\right) + 0(\sigma^2 h) + 0(\sigma^4)}$$

2.méthode en arbre

$$\mathcal{P}'(1) = \frac{(q-1)^2}{(q^n-1)^2}\left[(p+1)\sum_{j=0}^{n-1} q^{2j} + 2\sum_{\delta=1}^{P}\delta \sum_{(i,j)\in \phi_1^{-1}(\delta)} q^i q^j\right]$$

On écrit i et j sous la forme :

$$i = 2C\,2^{p-\delta} + u$$
$$j = (2C+1)\,2^{p-\delta} + v$$

avec $0 \leqslant C \leqslant 2^{\delta-1} - 1 \quad 0 \leqslant u, v \leqslant 2^{p-\delta}-1$

On a alors :

$$\sum_{(i,j)\in\phi^{-1}(\delta)} q^i q^j = q^{2^{p-\delta}} \left(\sum_{C=0}^{2^{\delta-1}-1} q^{C2^{p-\delta+2}}\right)\left(\sum_{u=0}^{2^{p-\delta}-1} q^u\right)\left(\sum_{v=0}^{2^{p-\delta}-1} q^v\right)$$

$$= q^{2^{p-\delta}} \left(\frac{q^{2^{p-\delta}}-1}{q-1}\right)^2 \sum_{C=0}^{2^{\delta-1}-1} \left(q^{2^{p-\delta+2}}\right)^C$$

$$= q^{2^{p-\delta}} \left(\frac{q^{2^{p-\delta}}-1}{q-1}\right)^2 \left(\frac{q^{2^{p+1}}-1}{q^{2^{p-\delta+2}}-1}\right)$$

d'où

$$\mathcal{P}'(1) = \frac{(q-1)^2}{(q^n-1)^2}\left[(p+1)\frac{q^{2n}-1}{q^2-1} + 2\sum_{\delta=1}^{P}\delta\, q^{2^{p-\delta}} \left(\frac{q^{2^{p-\delta}}-1}{q-1}\right)^2 \left(\frac{q^{2^{p+1}}-1}{q^{2^{p-\delta+2}}-1}\right)\right]$$

$$= (p+1)\frac{q-1}{q+1}\,\frac{q^n+1}{q^n-1} + 2\,\frac{q^n+1}{q^n-1}\sum_{\mu=0}^{p-1}(p-\mu)\, q^{2^\mu}\,\frac{(q^{2^\mu}-1)^2}{q^{2^{\mu+2}}-1}$$

$$= \frac{q^n+1}{q^n-1}\left[(p+1)\frac{q-1}{q+1} + 2\sum_{\mu=0}^{p-1}(p-\mu)\, q^{2^\mu}\,\frac{q^{2^\mu}-1}{(q^{2^\mu}+1)(q^{2^{\mu+1}}+1)}\right]$$

Il suffit ici d'étudier deux cas : en effet, $\mathcal{P}'(1)$ est le même pour q et 1/q (voir la deuxième remarque du paragraphe II.2.1).

- cas 1 : $q \gg 1$

comme :

$$q^{2^\mu} \cdot \frac{q^{2^\mu}-1}{(q^{2^\mu}+1)(q^{2^{\mu+1}}+1)} = \frac{1}{q^{2^\mu}} - \frac{2}{q^{2^{\mu+1}}} + 0\left(\frac{1}{q^{3.2^\mu}}\right)$$

on obtient l'estimation suivante :

$$\mathcal{P}'(1) = (p+1)\frac{q-1}{q+1} + 2\left(\sum_{\mu=0}^{1}(p-\mu)\frac{1}{q^{2\mu}} - \frac{2p}{q^2}\right) + 0\left(\frac{1}{q^3}\right)$$

$$= (p+1) + \frac{2(p-q)}{q(q+1)} - \frac{2(p+1)}{q^2} + 0\left(\frac{1}{q^3}\right)$$

d'où :

$$\boxed{V^a(1+\Lambda) = (p+1)\sigma^2 + \frac{2(p-q)}{q(q+1)}\sigma^2 - \frac{2(p+1)}{q^2}\sigma^2 + 0\left(\frac{\sigma^2}{q^3}\right) + 0(\sigma^4)}$$

- cas 2 : $q = 1 + h \quad 0 \leqslant h \ll 1$

$$\frac{q^n+1}{q^n-1} = \frac{2+nh}{nh} + O(h)$$

$$q^{2^\mu}\frac{q^{2^\mu}-1}{(q^{2^\mu}+1)(q^{2^{\mu+1}}+1)} = \frac{(1+2^\mu h)2^\mu h}{2(2+2^\mu h)(1+2^\mu h)} + O(h)$$

$$= \frac{2^\mu h}{2(2+2^\mu h)} + O(h)$$

d'où :

$$\mathcal{P}'(1) = \frac{2+nh}{n}\frac{p+1}{2+h} + \frac{2+nh}{nh}\sum_{\delta=1}^{P}\delta\frac{nh/2^\delta}{2+nh/2^\delta} + O(h)$$

$$= \frac{2+nh}{n}\frac{p+1}{2+h} + (2+nh)\sum_{\delta=1}^{P}\delta\frac{1}{2^{\delta+1}+nh} + O(h).$$

Soit $k(n) = \sum_{\delta=1}^{P}\delta\frac{1}{2^{\delta+1}+nh}$

d'où :

$$\boxed{V^a(1+\Lambda) = (2+nh)\left[\frac{p+1}{2n} + k(n)\right]\sigma^2 + O(h\sigma^2) + 0(\sigma^4)}$$

k(n) est majoré par la constante obtenue pour h = 0, c.a.d. :

$$0 < k(n) \leqslant 1 - \frac{p+2}{2n} < 1$$

Lorsque nh<<1, on retrouve ici les résultats obtenus dans le cumul de termes de même grandeur, cas correspondant à h = 0. Lorsque nh>>1, $\mathcal{P}'(1)$ se comporte comme nk(n)h. En effet :

$$\mathcal{P}'(1) = (2+nh)\left[\frac{p+1}{2n} + k(n)\right] + O(h)$$

3.comparaison des deux méthodes

Le rapport r_σ de l'écart-type de l'erreur relative commise par la méthode en ligne par l'écart-type de l'erreur relative commise par la méthode en arbre donne, suivant les valeurs de q :

* $q \gg 1$

$$r_\sigma \sim \sqrt{2/\log_2 n}$$

* $q \ll 1$

$$r_\sigma \sim \sqrt{n/\log_2 n}$$

* $q = 1 + h \quad |h| \ll 1$

$$r_\sigma \sim \sqrt{n/6} \qquad \text{lorsque } n|h| \ll 1$$

$$r_\sigma \sim \sqrt{n/3(2+nh)\left(\frac{p+1}{2n} + k(n)\right)} \quad \text{lorsque } n|h| \gg 1 \;;\; 0 < k(n) < 1 - \frac{\log_2 n + 2}{2n}$$

Les courbes donnant le rapport r_σ en fonction de n, pour différentes valeurs de q sont données à l'annexe 1.

Lorsque q >> 1, c.a.d. lorsque la suite des termes à cumuler croit très rapidement, la méthode en ligne est supérieure à la méthode en arbre. Dans les autres cas, la précision obtenue par la méthode en arbre est meilleure.

On a ici raisonné sur les valeurs probables de l'erreur relative. Dans (LA 80) est exposée une méthode permettant d'estimer la borne supérieure de ces erreurs. Cette méthode est décrite à l'annexe 2. Pour les applications étudiées, notre approche donne le même classement des deux algorithmes, ligne et arbre, que celui obtenu à partir des bornes supérieures des erreurs, mais avec des différences moins marquées.

II.3. Cumul de termes à signe variable

Dans ce cas on ne peut plus étudier l'erreur relative car il est possible de rencontrer des sommes arbitrairement petites. Malheureusement la situation est plus difficile à modéliser que dans le cas où le signe est constant et on se limite donc à l'étude du cas où chaque terme de la somme suit une loi uniforme sur un même intervalle centré que l'on ramène à (-a, +a).

Soit $\sigma_n = a(\frac{\sqrt{n}}{3})$ l'écart type de la somme de n v.a. 2 à 2 indépendantes et uniformément réparties sur (-a , +a) . On admet alors le principe suivant :

H2 — A partir des v.a. x et y, où x est somme de n v.a. 2 à 2 indépendantes uniformément réparties sur (-a , +a) et y somme de p de variables de même type
On définit la variable aléatoire ε par
$$fl(x+y) = x+y+\varepsilon\,\sigma_{n+p}$$
La variable aléatoire ε est centrée et de variance σ^2 indépendante de n et p.

On peut maintenant estimer l'écart type de l'erreur absolue sur le résultat du cumul de n termes $(x_i)_{i=0,n-1}$ uniformément répartis sur (-a, +a) et deux à deux indépendants et cela pour les deux méthodes étudiées.

1.méthode en ligne

Le principe (H2) permet d'écrire le schéma sous la forme suivante

$$s_0 = fl(x_0) = x_0 + \varepsilon_0\,\sigma_1$$

$$s_r = fl(x_{r-1} + x_r) = s_{r-1} + x_r + \varepsilon_r\,\sigma_{r+1}$$

où les v.a. ε_i sont 2 à 2 indépendantes centrées et de même variance σ^2. L'écart type de l'erreur absolue $\sum_{i=0}^{n-1} \sigma_{i+1}\,\varepsilon_i$ sur le résultat est donc :

$$\sigma(\text{erreur "arbre"}) = \frac{a\sigma}{\sqrt{3}}\sqrt{\sum_{0}^{n-1}(i+1)} = \frac{a\,\sigma\sqrt{n(n+1)}}{\sqrt{6}}$$

2.méthode en arbre ($n=2^p$)

Dans ce cas le schéma devient :

$$\begin{cases} s_0^i = fl(x_i) = x_i + \varepsilon_0^i\,\sigma_1 & i = 0,\ n-1 \\ s_r^i = fl(s_{r-1}^{2i} + s_{r-1}^{2i+1}) = s_{r-1}^{2i} + s_{r-1}^{2i+1} + \varepsilon_r^i\,\sigma_{2r} & r=1,p;\ i=0,\ \frac{n}{2^r}-1 \end{cases}$$

où les v.a. $\mathcal{E}_r^i$ sont encore deux à deux indépendantes, centrées et de même variance σ^2. L'écart type de l'erreur absolue :

$$\sum_{n=o}^{p} \sigma_{2r} \left(\sum_{i=o}^{n/2^{r-1}} \mathcal{E}_r^i \right)$$

est alors :

$$\sigma(\text{erreur "arbre"}) = \frac{a\sigma}{\sqrt{3}} \sqrt{\sum_{r=o}^{p} 2^r \cdot \frac{n}{2^r}} \quad \frac{a\sigma}{\sqrt{3}} \sqrt{n(p+1)}$$

3.comparaison des deux méthodes

Le rapport des écarts-type de l'erreur commise par chacune des deux méthodes est :

$$r_\sigma = \sigma(\text{erreur "ligne"})/\ \sigma(\text{erreur "arbre"})$$

$$r_\sigma = \sqrt{\frac{n+1}{2(\log_2 n+1)}}$$

Ce rapport fait encore apparaitre l'amélioration apportée par le méthode en "arbre" sur la méthode en "ligne". La courbe donnant le rapport r_σ en fonction de n est donnée à l'annexe 1.

application numérique

L'application est faite sur le calculateur CII-HB DPS 8 - Système Multics. Un mot simple précision possède 27 bits de mantisse ($-127 \leqslant e \leqslant 127$). Pour la valeur expérimentale $\sigma = 0.25.\ 10^{-8}$

On trouve les valeurs suivantes lorsque $n=2^{17} = 131\ 072$

σ (erreur "ligne") $\simeq 1.3.\ 10^{-4}$

σ (erreur "arbre") $\simeq 2.2\ \ 10^{-6}$

$r_\sigma \simeq 60$

II.4. Application au produit scalaire

Toute l'étude précédente s'applique naturellement à l'étude du produit scalaire. En effet, pour calculer la quantité

$$s = \sum_{i=0}^{n-1} x_i\ y_i$$

il suffit de considérer que l'erreur commise sur le terme général $x_i y_i$ de la somme correspond à l'erreur de la multiplication flottante.

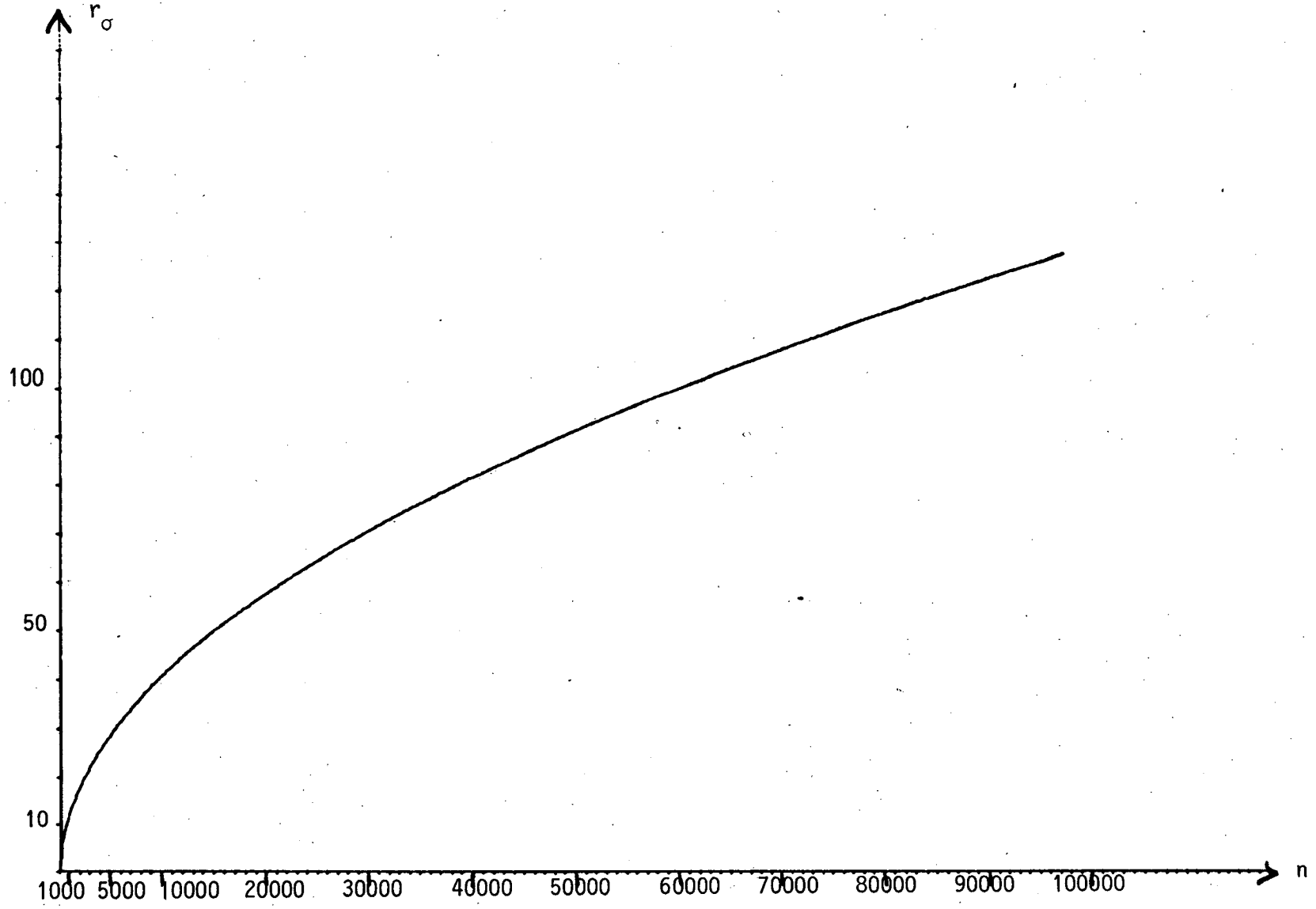
CONCLUSION

Dans cette étude on a confirmé ce qui était déjà connu à savoir la supériorité, pour la précision du cumul d'un grand nombre de termes, de la méthode en "arbre" sur la méthode en "ligne" (cette supériorité n'a pu être mise en défaut que dans le cas où les termes forment une suite croissante de nombres positifs).

Cependant on montre ici que le gain réel apporté n'est pas aussi important que ce que laisse prévoir une étude directe réalisée à partir des majorations d'erreurs ; par exemple dans le cas du cumul de termes de même signe et de même ordre de grandeur le rapport des erreurs est de $\sqrt{\frac{n}{6}}$ au lieu de $\frac{n}{\log_2 n}$. Ainsi il semble difficile d'avoir une amélioration de la précision qui soit supérieure à un rapport 10^3, amélioration qui correspondrait à rallonger la longueur du mot d'une dizaine de bits. Ceci reste très inférieur au gain apporté par le passage en double précision.

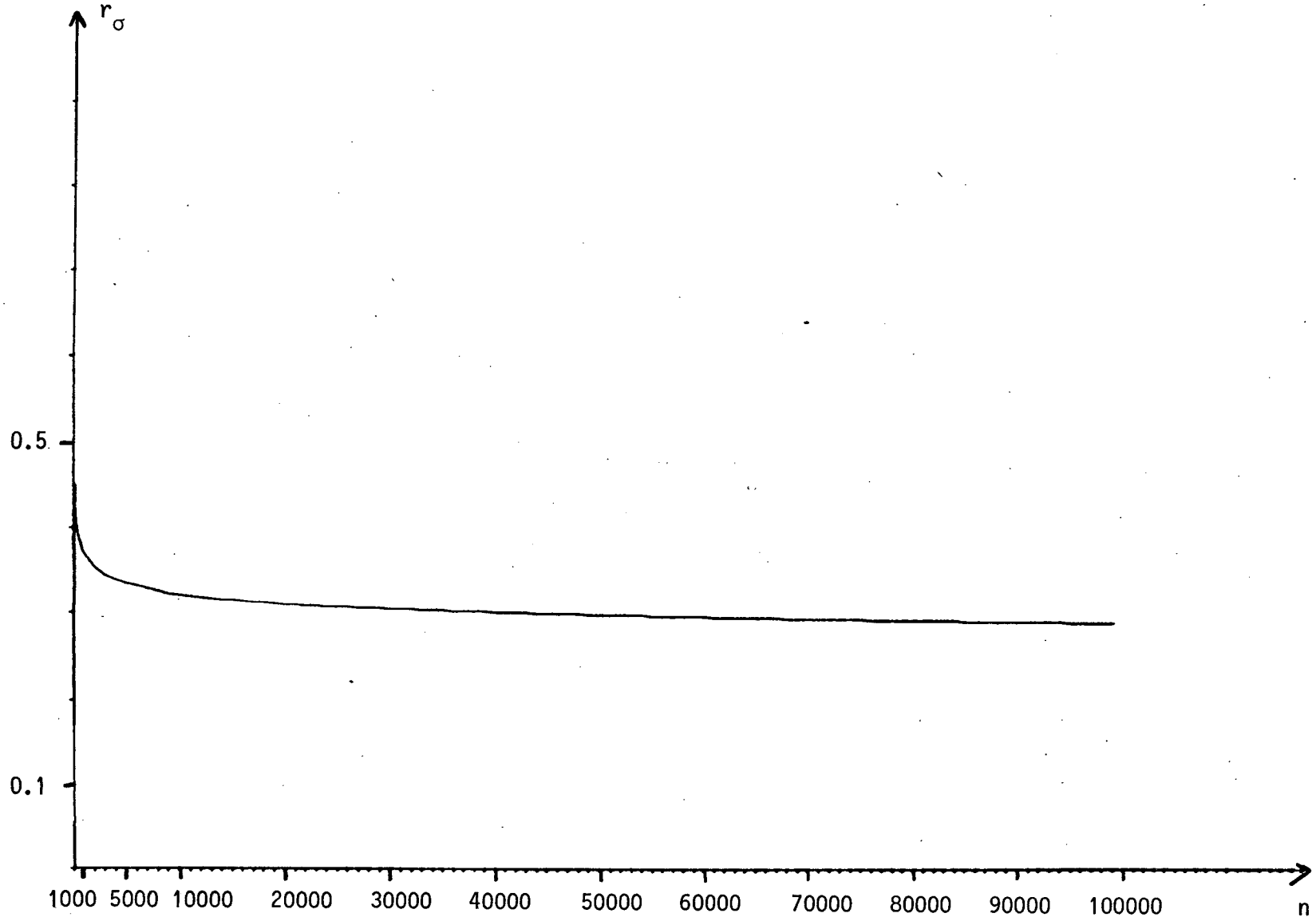
Ce gain peut pourtant être satisfaisant dans de nombreuses situations. Il serait donc intéressant d'avoir dans l'ensemble des programmes d'algèbre linéaire élémentaire disponibles sur un calculateur, un programme de la méthode en "arbre". Correctement écrit en assembleur la pénalité à payer en temps, par rapport à la méthode en "ligne", serait faible. On trouvera en annexe l'algorithme permettant de programmer la méthode en "arbre".

ANNEXE 1



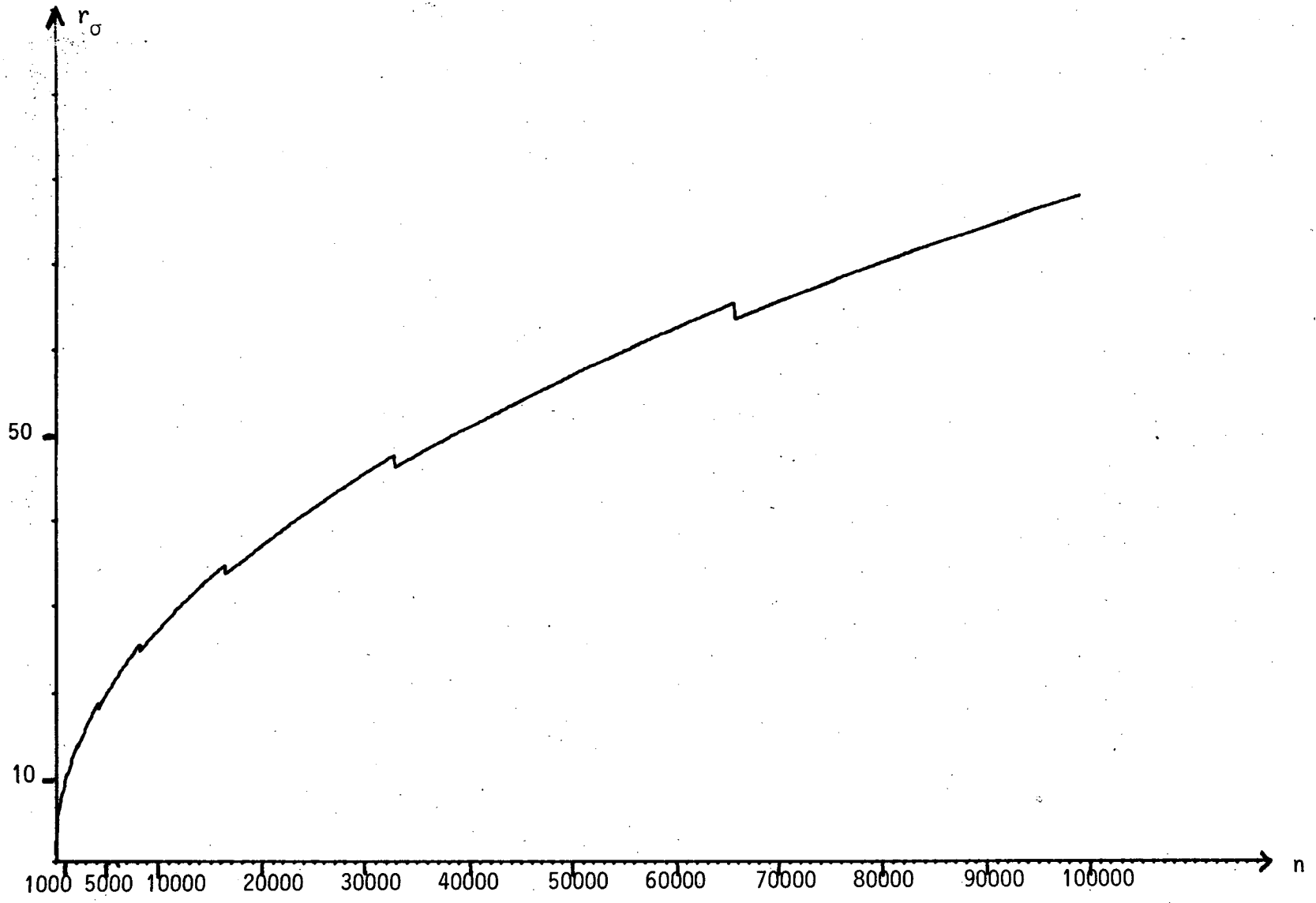
- Termes de même grandeur

Rapport:écart-type (erreur relative, méthode ligne)/écart-type (erreur relative, méthode arbre)



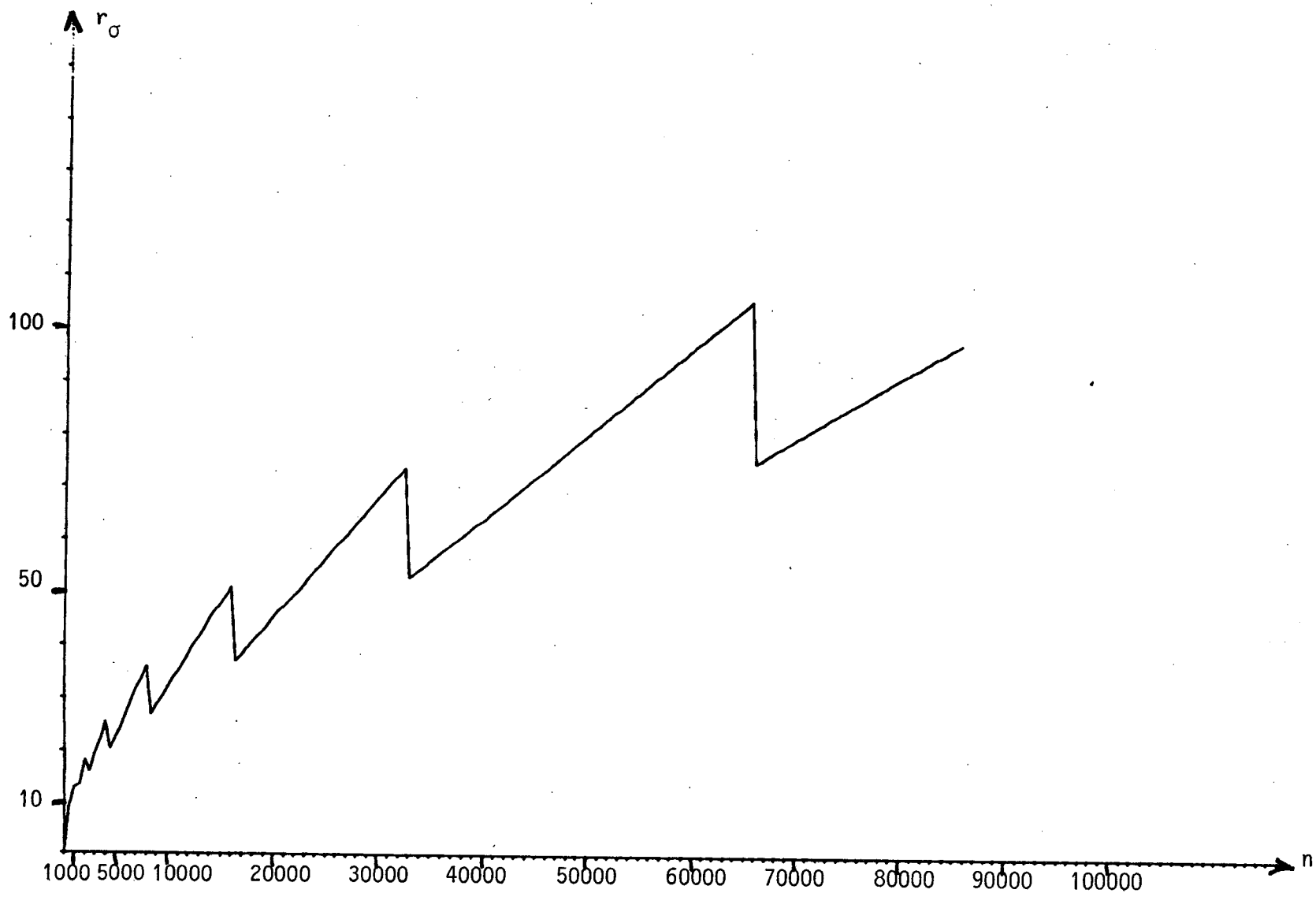
- Termes en progression géométrique q = 3

Rapport : écart-type (erreur relative, méthode ligne)/écart-type (erreur relative, méthode arbre)



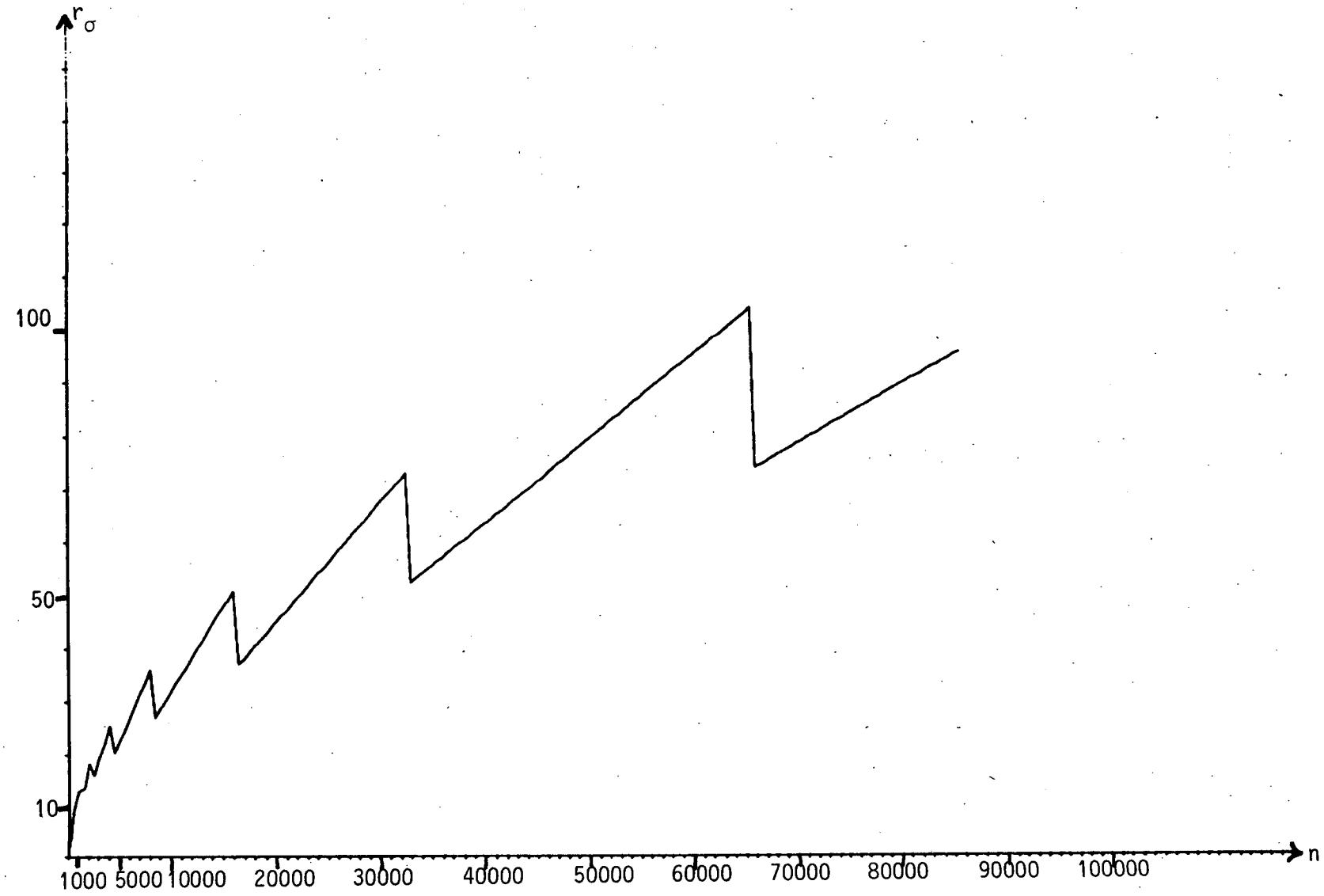
- Termes en progression géométrique q = 0.3333

Rapport : écart-type (erreur relative, méthode ligne)/ écart-type (erreur relative, méthode arbre)



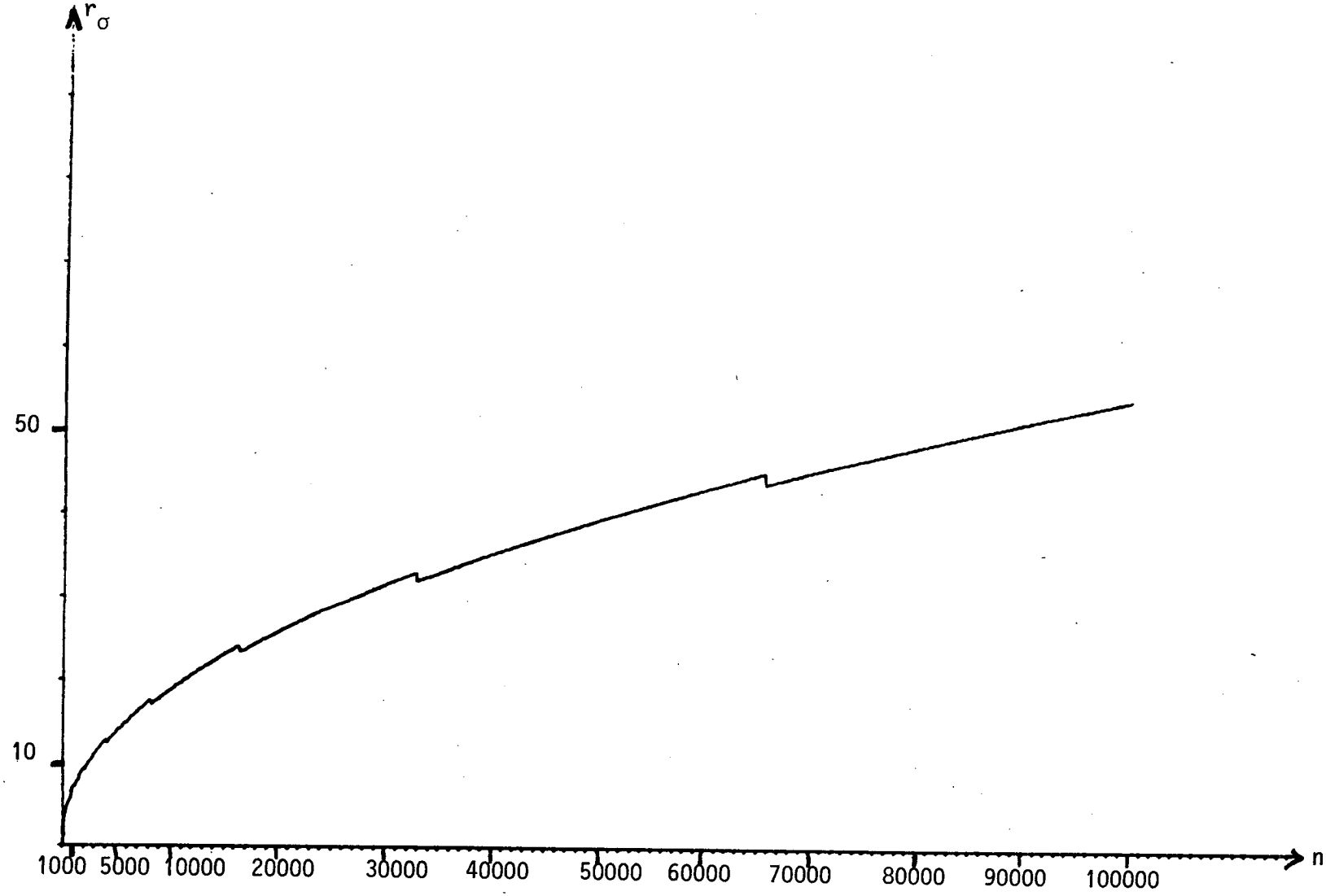
- Termes en progression géométrique q = 0.999999

Rapport : écart-type (errreur relative, méthode ligne)/ écart-type (erreur relative, méthode arbre)



- Termes en progression géométrique. q = 1.000001

Rapport : écart-type (erreur relative, méthode ligne)/écart-type (erreur relative, méthode arbre)



- Termes de signes variables

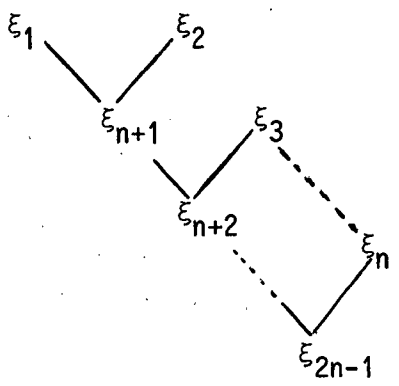
Rapport : écart-type (erreur absolue, méthode ligne)/écart-type (erreur absolue, méthode arbre)

ANNEXE 2

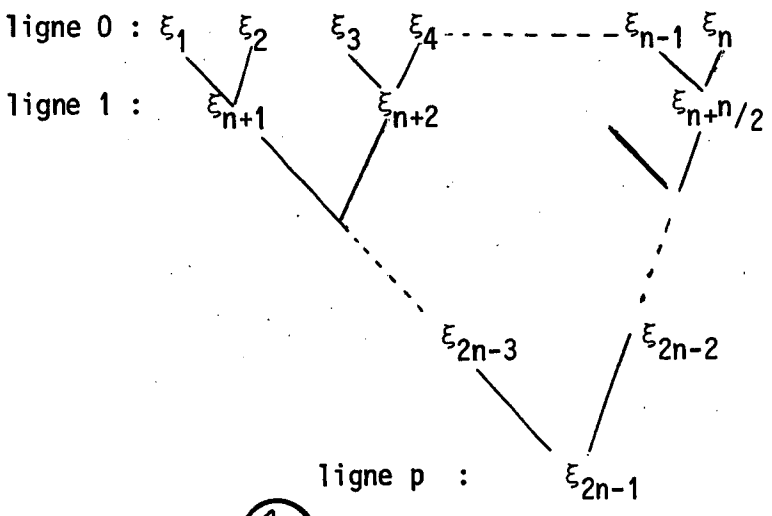
STABILITE DES ALGORITHMES (FORMALISATION LARSON-SAMEH)

Pour évaluer la stabilité des deux algorithmes on adopte la formulation développée dans (LA 80).

Les deux algorithmes, soit cumul en ligne et cumul en arbre, peuvent se décrire par les 2 graphes $\mathcal{A}1$ et $\mathcal{A}2$ suivants :



$\mathcal{A}1$



$\mathcal{A}2$

où $\xi_i = x_{i-1}$ pour i = 1,n , $s = \xi_{2n-1}$ avec $n = 2^p$.

Chaque noeud (ξ_{n+i}) où i = 1, n-1 correspond à une variable somme des deux variables origines des arcs dont il est l'extrêmité.

La formulation de Larson et Sameh consiste à linéariser l'erreur relative commise sur le résultat :

soit $\varepsilon = 2^{-t}$ l'erreur relative locale maximale commise sur un noeud ξ_1 l'erreur locale commise s'écrit alors $(\rho\xi_1)_L.\varepsilon$ où $|(\rho\xi_1)_L| \leqslant 1$.

On recherche l'erreur relative globale commise sur le résultat $s = \xi_{2n-1}$: d'après (LA 80) elle est égale à :

$$(\rho\,\xi_{2n-1})_T\,\varepsilon = (\overline{A} \mid \overline{B}) \cdot \begin{bmatrix} (\rho\xi_1)_L \\ (\rho\xi_n)_L \\ (\rho\xi_{n+1})_L \\ (\rho\xi_{2n-1})_L \end{bmatrix} \varepsilon \ .$$

où $\overline{A}$ et $\overline{B}$ sont les matrices-ligne suivantes :

$$\overline{A} = \left[\frac{\xi_1}{\xi_{2n-1}} \quad , \quad \frac{\xi_2}{\xi_{2n-1}} \quad , \ \ldots \ , \quad \frac{\xi_n}{\xi_{2n-1}} \right]$$

$$\overline{B} = \left[\frac{\xi_{n+1}}{\xi_{2n-1}} \quad , \ \ldots \ , \quad \frac{\xi_{2(n-1)}}{\xi_{2n-1}} \quad , \ 1 \right]$$

La matrice $\overline{A}$ fait intervenir les erreurs commises sur les données, par contre $\overline{B}$ fait intervenir les erreurs commises lors des calculs dans l'algorithme.

Soit $d = \begin{bmatrix} (\rho\xi_1)_L \\ \vdots \\ (\rho\xi_n)_L \end{bmatrix}$, $g = \begin{bmatrix} (\rho\xi_{n+1})_L \\ \vdots \\ (\rho\xi_{2n-1})_L \end{bmatrix}$ et B_r la boucle unité de $\mathbb{R}^r$

pour la norme $L\infty$.

On a donc la relation :

$$[\rho\xi_{2n-1}]_T = \overline{A}\, d + \overline{B}\, g \qquad \text{avec } d \in B_n \text{ et } g \in B_{n-1}$$

On recherche le plus petit θ réel positif tel que $\overline{B}(B_{n-1}) \subset \theta . \overline{A}(B_n)$. Pour cette valeur de θ, et pour une valeur du résultat obtenu à partir de données exactes, on aurait obtenu le même résultat en calculant sans erreur mais à partir de données initiales perturbées $\xi_i\ (1 + (\rho\xi_i)_L\ \varepsilon)$ $i = 1, n$ où $|(\rho\xi_i)| \leqslant \theta$. Cela correspond à l'analyse d'erreur nommée "Backward Analysis".

Ici l'algorithme ne produit qu'un seul résultat donc $\overline{A}(B_n)$ et $\overline{B}(B_{n-1})$ sont deux intervalles $(-a, a)$ et $(-b, b)$ ce qui entraine que $\theta = \frac{b}{a}$.

De plus les éléments de la somme étant positifs on a :

$$\| \overline{A} \|_\infty = \sum_{i=1}^{n} \frac{\xi_i}{\xi_{2n-1}} = 1 \text{ donc } a \leqslant 1 \text{ ; or } \overline{A} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = 1 \text{ d'où } a = \|\overline{A}\|_\infty = 1$$

de même $\| \overline{B} \|_\infty = b$.

On en déduit $\theta = \|\overline{B}\|_\infty$

Pour comparer les deux algorithmes on calcule donc $\|B\|_\infty$ pour chacun d'eux dans les deux cas de l'étude faite du point de vue probabiliste.

On remarque d'abord que pour l'algorithme $\mathcal{A}_2$, la somme des éléments de chaque ligne est constamment égale à ξ_{2n-1}. Cela entraine que dans tous les cas :

$$\theta_2 = \|\overline{B}_2\| = p = \log_2 n$$

Pour l'algorithme $\mathcal{A}_1$, il faut distinguer les cas.

1er cas : termes de même grandeur : $\xi_i = \frac{1}{n}$ $i = 1,n$; $\xi_{2n-1} = 1$

$$\|\overline{B}_1\| = \sum_{i=1}^{n-1} \frac{i+1}{n} = \frac{(n-1)(n+2)}{2n}$$

$$\theta_1 = \frac{n+1}{2} + 0\left(\frac{1}{n}\right)$$

2ème cas : termes en progression géométrique : $\xi_i = q^{i-1} \cdot \frac{q-1}{q^n-1}$; $\xi_{2n-1} = 1$

$$\|\overline{B}_1\| = \left[\sum_{i=1}^{n-1} \left(\sum_{j=1}^{i+1} q^{j-1}\right)\right] \frac{q-1}{q^n-1}$$

$$\|\overline{B}_1\| = \frac{q^{n+1} - q^2 - (n-1)q + (n-1)}{(q^n-1)(q-1)}$$

On évalue cette quantité suivant l'ordre de grandeur de q :

si $q \gg 1$ $\quad \theta_1 = 1 + \frac{1}{q} + 0\left(\frac{1}{q^2}\right)$

si $0 \leqslant q \ll 1$ $\quad \theta_1 = (n-1) + 0(q^2)$

si $q = 1 + h$ avec $|h| \ll 1$: $\quad \theta_1 = \frac{(n-1)(n+2)}{2n} + 0(nh)$

		algorithme $\mathcal{A}_1$	algorithme $\mathcal{A}_2$
termes de même grandeur		$\frac{n}{2}$	$\log_2 n$
termes en progression de raison q	$q \gg 1$	$1 + \frac{1}{q}$	$\log_2 n$
	$0 \leqslant q \ll 1$	n	$\log_2 n$
	$q = 1 + h$ $\quad \|h\| \ll 1$	$\frac{n}{2}$	$\log_2 n$

Tableau : valeurs approchées du paramètre θ dans chaque cas étudié pour un cumul de termes positifs.

ANNEXE 3

ALGORITHME POUR UN CUMUL PAR LA METHODE "EN ARBRE"

La somme à calculer est $S_{n-1} = \sum_{i=0}^{n-1} x_i$ où n est un entier naturel quelconque non nul.

Il est nécessaire d'utiliser un tableau s qui mémorisera pour j fixé entre 1 et n les sommes partielles correspondant à l'écriture binaire de j :

pour $j = \overline{j_{l-1} \cdots j_0}^2$ avec $j_{l-1} = 1$

on introduit les quantités suivantes définies par récurrence :

$$J_{l-1} = j_{l-1}\, 2^{l-1} \quad \text{et} \quad J_k = J_{k+1} + j_k . 2^k \text{ pour } k = l-2, \ldots, 0$$

et les sommes partielles suivantes

$$s_1 = \sum_{i=0}^{J_{l-1}-1} x_i \; ; \; s_2 = \sum_{i=J_{l-1}}^{J_{l-2}-1} x_i \; ; \ldots ; \; s_l = \sum_{i=J_1}^{J_0-1} x_i$$

avec la convention qui si une somme porte sur un nombre nul de termes elle est nulle.

La longueur du tableau s est $\lceil \log_2 j \rceil$. Si n n'est pas une puissance de 2 il faudra, après la dernière étape, cumuler toutes les sommes partielles, ce que l'on fera séquentiellement dans le sens décroissant des indices, afin d'être optimal dans le cas où les éléments sont de même signes et de grandeurs comparables.

Pour l'algorithme, on remplace le tableau s par sa forme condensée c'est à dire en ne considérant que les sommes qui portent sur un nombre non nul de termes ; dans ce cas l ne sera plus $\lceil \log_2 j \rceil$ mais le nombre de bits à 1 dans la décomposition binaire de j.

```
début
  n,p : constante-entière ; co p = ⌈log₂ n⌉ fco
  s : tableau (1 : p) de reels ;
  x (entier : i) : fonction reel ;
  i, m, l, l1 : entier ;
  som : reel ; co contient en sortie le cumul fco

  l = 0 ;
boucle 1 :
  pour i = 0, n - 1
      faire
        m = i + 1 ;
        l = l + 1 ;
        s(l) = x(i) ;
        tant que m ≡ 0 (2)
              faire
                l = l - 1 ;
                s(l) = s(l) + s (l+1) ;
                m = m/2
              fait
      fait ;
  l1 = l ;
  som = s (l1) ;
boucle 2 :
  tant que l1 ≠ 1
      faire
        l1 = l1 - 1 ;
        som = som + s (l1)
      fait
fin.
```

Dans le cas où chaque terme x_k dépend du cumul $S_{k-1} = \sum_{i=0}^{k-1} x_i$, l'algorithme précédent n'est pas applicable. En effet il faut à chaque étape, cumuler les valeurs mémorisées dans le tableau s, ce qui revient à insérer la boucle 2 à la fin du corps de la boucle 1. Pour évaluer le nombre d'additions supplémentaires que cela entraine, on se place dans le cas où $n = 2^p$.

A l'étape i de la boucle 1 on effectue l-1 additions supplémentaires où l est le nombre de bits à 1 dans la décomposition binaire de (i+1). On introduit donc au total $2^p = n$ additions supplémentaires. Il faut cependant remarquer que ce doublement du nombre des additions à effectuer ne correspond pas à un doublement du volume des calculs. En effet l'évaluation d'un terme coutera sans doute plus cher qu'une simple addition.

ANNEXE 4

FORMULES UTILES

$$\sum_{i=o}^{n} i = \frac{n(n+1)}{2}$$

$$\sum_{i=o}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\sum_{i=o}^{n} a^i = \frac{a^{n+1}-1}{a-1}$$

$$\sum_{i=1}^{n} i\, a^i = \frac{n\, a^{n+2} - (n+1)\, a^{n+1} + a}{(a-1)^2}$$

$$\sum_{i=o}^{n-1} (n-i)a^i = \frac{a^{n+1} - (n+1)a + n}{(a-1)^2}$$

$$(1+x)^{\alpha} = 1 + \sum_{k=1}^{\infty} \frac{\alpha(\alpha-1)\ldots(\alpha-k+1)}{k!}\, x^k$$

BIBLIOGRAPHIE

(FE 82) FELDSTEIN A. ; GOODMAN R. : "Loss of significance in floating point substraction and addition".
IEEE Transactions on Computers. Vol C. 31 n° 4, April 1982, pp 328-335

(KU 77) KUCK D.J. ; PARKER D.S. ; SAMEH A.H. : "Analysis of rounding methods in floating point arithmetic".
IEEE Transactions on Computers. Vol C.26 n° 7, July 1977, pp 643-650

(KU 81) KULISCH U.W. ; MIRANKER W.L. : "Computer Arithmetic in theory and practice".
Academic Press, 1981.

(LA 80) LARSON J.L. ; SAMEH A.H. : "Algorithms for roundoff error analysis- A relative error approach".
Computing n° 24. 1980. pp 275-297

(TS 74) TSAO N. : "On the distributions of significant digits and roundoff errors".
Communications of the ACM Vol. 17 n°5, May 1974. pp 269-271

(WI 63) WILKINSON J.H. : "Rounding errors in algebraic processes".
Notes on applied science n° 32 - Imprimé et publié par :
Her Majesty's Stationery Office. 1963.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique